Nicht für den Prüfling bestimmt

I Erläuterungen

Voraussetzungen gemäß KCBG und Abiturerlassen BG jeweils in der für den Abiturjahrgang geltenden Fassung

Standardbezug
Die nachfolgend ausgewiesenen Kompetenzbereiche sind für die Bearbeitung der jeweiligen Aufgabe besonders bedeutsam. Darüber hinaus können weitere, hier nicht explizit ausgewiesene Kompetenzen für die Bearbeitung der Aufgabe nachrangig bedeutsam sein, zumal die Kompetenzen in engem Bezug zueinander stehen. Die Operationalisierung des Bezugs zu den Kompetenzbereichen des Standardbezugs erfolgt in Abschnitt II.

Aufgabe	Kompetenzen				
	K1	K2	K3	K4	K5
1.1	X				
1.2	X				
1.3		X			
1.4				X	
1.5				X	
1.6	X				
1.7					X
2.1		X			
2.2		X			
2.3			X		
2.4			X		
2.5			X		
3.1			X		
3.2			X		
3.3					X
3.4				X	
3.5					X

Inhaltlicher Bezug
Die nachfolgend ausgewiesenen Themenfelder sind die wesentliche inhaltliche Grundlage für die vorliegenden Aufgaben. Darüber hinaus können weitere, hier nicht explizit ausgewiesene Themenfelder für die Bearbeitung nachrangig bedeutsam sein.
Q1: Objektorientierte Softwareentwicklung
Q2: Digitale Steuerungstechnik
Q3: Prozessautomatisierung
verbindliche Themenfelder: Objektmodellierung (Q1.1), Implementierung von Klassen und ihren Beziehungen (Q1.2), Such und Sortieralgorithmen (Q1.3), Synthese statischer und sequentieller Logikschaltungen (Q2.1), Mikrocontroller (Q2.2), Einführung in die Prozessautomatisierung (Q3.1)

II Lösungshinweise

In den nachfolgenden Lösungshinweisen sind alle wesentlichen Gesichtspunkte, die bei der Bearbeitung der einzelnen Aufgaben zu berücksichtigen sind, konkret genannt und diejenigen Lösungswege aufgezeigt, welche die Prüflinge erfahrungsgemäß einschlagen werden. Selbstverständlich sind jedoch Lösungswege, die von den vorgegebenen abweichen, aber als gleichwertig betrachtet werden können, ebenso zu akzeptieren.

Nicht für den Prüfling bestimmt

Aufg.	erwartete Leistungen	BE I	BE II	BE III
1.1	entwickeln, bestimmen 			
	entwickeln		3	2
	bestimmen	5		
1.2	beschreiben, erläutern Die Methode `main` befindet sich in der Klasse `FuhrparkIO`. Die Klasse `Kunde` kann beliebig viele Buchungen besitzen. Dieses wird durch die unidirektionale Assoziation mit dem Namen `besitzt` und der Multiplizität 0..* in Richtung `Buchung` dargestellt. In der Klasse `Kunde` befindet sich die Referenzvariablenliste `myBuchungen` mit dem Datentyp `Buchung`, in die mithilfe der Methode `neueBuchung()` neue Buchungen zu der Liste hinzugefügt werden können. Als Parameter besitzt diese Methode eine `buchung` vom Datentyp `Buchung`. Die Klasse `Buchung` enthält eine Referenzvariablenliste `warenkorb`, in der die gebuchten Waren gespeichert werden. Diese werden mithilfe der Methode `neueWare()` zu der Liste hinzugefügt. Als Parameter besitzt diese Methode ein item vom Datentyp `Ware`. Es sind noch zwei weitere Methoden angegeben: Der Getter für das Attribut `dauer` mit dem Rückgabedatentyp `int` und die Methode `angebotBerechnen` mit dem Rückgabedatentyp `double`. Zwischen der Klasse `Buchung` und der abstrakten Klasse `Ware` existiert die bidirektionale Assoziation `enthält`. Eine Buchung kennt beliebig viele Waren, aber eine Ware kennt nur eine Buchung. Die Referenzvariable `myBuchung` für die Buchung in der Klasse `Ware` ist vom Datentyp `Buchung` und die Sichtbarkeit ist			

Nicht für den Prüfling bestimmt

Aufg.	erwartete Leistungen	BE I	BE II	BE III
	protected. In der Klasse `Ware` gibt es den Setter für die Referenzvariable `myBuchung` mit dem Parameter `myBuchung` mit dem Datentyp `Buchung` und die Methode `myKosten` mit dem Rückgabedatentyp `double`. Von der Klasse `Ware` erben die drei Klassen `Anhaenger`, `Traktor` und `Saatgut (Vererbungs-Beziehung)`. In allen drei Klassen gibt es die Methode `myKosten` mit dem Rückgabedatentyp `double`. Von der abstrakten Klasse `Anhaenger` erben die beiden Klassen `SaatgutVerteiler` und `TransportAnhaenger`.			
	beschreiben	7		
	erläutern		3	
1.3	entwickeln, ergänzen :Kunde — name="Heimeler" userName="heimlich" Password="asdofjwe9" IBAN="DE123456789" :Buchung — datum="27.7.2021" dauer=2 :SaatgutVerteiler — saatkapazitaet=6000 kostenProTag=180. :Traktor — hersteller="Schwerlast" leistung=700. kostenProTag=450. :Saatgut — menge=20. kostenProZentner=130.			
	entwickeln		1	3
	ergänzen		2	
1.4	implementieren (siehe Code unten)		3	6

```
import java.util.*;
public class Buchung {
  private String datum;
  private int dauer;
  private ArrayList<Ware> warenkorb = new ArrayList<Ware>();
  private double angebotsPreis;
  public void neueWare(Ware item){
    warenkorb.add(item);
  }
  public double angebotBerechnen() {
  }
  public Buchung(String datum, int dauer) {
    this.datum=datum;
    this.dauer=dauer;
  }
  public int getDauer(){
    return(dauer);
  }
}

public abstract class Ware {
  protected Buchung myBuchung;
  protected void setMyBuchung(Buchung myBuchung){
    this.myBuchung=myBuchung;
```

Aufg.	erwartete Leistungen	BE I	BE II	BE III

	`}`
	`protected double myKosten(){}`
	`}`
	
	`public class Saatgut extends Ware {`
	`private double menge;`
	`private double kostenProZentner;`
	`public Saatgut(double menge, double kostenProZentner) {`
	`super();`
	`this.menge=menge;`
	`this.kostenProZentner=kostenProZentner;`
	`}`
	`public double myKosten(){}`
	`}`
	
	`public class Traktor extends Ware{`
	`private String hersteller;`
	`private double leistung;`
	`private double kostenProTag;`
	`public Traktor(String hersteller, double leistung, double kostenProTag) {`
	`super();`
	`this.hersteller=hersteller;`
	`this.leistung=leistung;`
	`this.kostenProTag=kostenProTag;`
	`}`
	`public double myKosten(){}`
	`}`			

1.5	implementieren
	`public double angebotBerechnen() {`
	`double kosten =0.;`
	`for (Ware item : warenkorb) {`
	`kosten=kosten+item.myKosten();`
	`}`
	`return kosten;`
	`}`
	`//Ware:`
	`protected double myKosten(){return(0.);}`
	`//Traktor:`
	`public double myKosten(){`
	`return(myBuchung.getDauer()*kostenProTag);`
	`}`
	`//Saatgut:`
	`public double myKosten(){`
	`return(menge*kostenProZentner);`
	`}`
			2	3

1.6	erläutern Von Polymorphie spricht man, wenn zwei Klassen denselben Methodennamen verwenden, die Implementierung der Methoden sich aber unterscheidet. In der abstrakten Klasse `Ware` ist die Signatur der Methode `myKosten()` aufgeführt. Von dieser Klasse erben u.a. die Klassen `Traktor` und `Saatgut`. In der Klasse `Traktor` findet die Berechnung in `myKosten()` mittels `dauer*kos-`	3	2	

Aufg.	erwartete Leistungen	BE		
		I	II	III
	`tenProTag` statt, während in der Klasse `Saatgut` die Berechnung in `myKosten()` mittels `menge*kostenProZentner` erfolgt. Man spricht in diesem Zusammenhang auch davon, dass die Methode `myKosten()` überschrieben wird.			
1.7	entwickeln, erläutern Unter der Klasse `Anhaenger` sind zwei Typen von Anhängern spezialisiert. Es gibt keine strukturellen Unterschiede zwischen den Anhängern und dem Traktor, so dass der Traktor auf die gleiche Ebene wie die Anhänger gestellt werden kann. Das bedeutet, dass die Klasse `Anhaenger` z.B. in `Fahrzeuge` umbenannt werden kann. Die Klasse `Traktor` würde dann von der Klasse `Fahrzeuge` erben, genauso wie die Klassen `SaatgutVerteiler` und `TransportAnhaenger`. Das Attribut `kostenProTag` ist schon in der jetzigen Klasse `Anhaenger` vorhanden, genauso wie die Behandlung der Kostenermittlung in der Methode `myKosten()`. Das heißt, dass sowohl das Attribut als auch die Methode im Sinne der Objektorientierung aus der Klasse `Traktor` entfernt werden. entwickeln erläutern	 1	 2	2
	Summe 50	**16**	**18**	**16**

Aufg.	erwartete Leistungen	BE		
		I	II	III
2.1	ergänzen (siehe Tabelle)	2		

N	D	E	V	S
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

Aufg.	**erwartete Leistungen**	**BE** I	**BE** II	**BE** III
2.2	ermitteln konjunktive Normalform: $S=(\bar{N}\vee D\vee E\vee V)\wedge(\bar{N}\vee\bar{D}\vee\bar{E}\vee\bar{V})$		3	
	disjunktive Normalform: $S=(\bar{N}\bar{D}\bar{E}\bar{V})\vee(\bar{N}\bar{D}\bar{E}V)\vee(\bar{N}\bar{D}E\bar{V})\vee(\bar{N}\bar{D}EV)\vee(\bar{N}D\bar{E}\bar{V})$ $\vee(\bar{N}D\bar{E}V)\vee(\bar{N}DE\bar{V})\vee(\bar{N}DEV)\vee(N\bar{D}\bar{E}V)$ $\vee(N\bar{D}E\bar{V})\vee(N\bar{D}EV)\vee(ND\bar{E}\bar{V})\vee(ND\bar{E}V)\vee(NDE\bar{V})$		3	
2.3	aufzeigen N / $\bar{N}$; D / $\bar{D}$; $\bar{V}$ / V / $\bar{V}$; $\bar{E}$ / E / $\bar{E}$ 1 1 1 1 1 0 1 1 1 1 1 1 0 1 1 1 $S=\bar{N}\vee(\bar{D}\wedge E)\vee(\bar{E}\wedge V)\vee(D\wedge\bar{V})$		2	2
2.4	ermitteln $S=\overline{\overline{\overline{N}}\wedge\overline{\bar{D}\wedge E}\wedge\overline{\bar{E}\wedge V}\wedge\overline{D\wedge\overline{\bar{V}}}}=\overline{N\wedge\overline{\bar{D}\wedge E}\wedge\overline{\bar{E}\wedge V}\wedge\overline{D\wedge\bar{V}}}$		3	
	zeichnen N D E V S &	3		

Nicht für den Prüfling bestimmt

Aufg.	**erwartete Leistungen**	**BE I**	**BE II**	**BE III**
2.5	herleiten, zeichnen $X=\overline{A\wedge B\wedge C\wedge D}=\overline{\overline{\overline{A\wedge B}}\wedge\overline{\overline{C\wedge D}}}$ herleiten zeichnen	 3	 2 1	
	Summe 24	**8**	**14**	**2**

Aufg.	**erwartete Leistungen**	**BE I**	**BE II**	**BE III**
3.1	implementieren LDI r16, 0b01111111 OUT 0x04, r16 LDI r17, 0b00000000 OUT 0x0A, r17		2	
3.2	entwickeln Solange nicht Störung Saatguttyp einlesen Start Särad? J / N Saatgut fein? J / N J: Schieber erhält Signale LOW HIGH N: Schieber erhält Signale HIGH LOW Solange nicht Stopp Särad Kurvenfahrt? J / N J: SOW_CURVE N: SOW_STRAIGHT Schieber zu (LOW LOW) Motor Särad aus (Alle vier Ausgangsbits LOW) Schieberendlage "zu" erreicht und Saatgut ablassen? J / N J: Bodenklappe öffnen WAIT_30s Bodenklappe schließen		6	6

Aufg.	**erwartete Leistungen**	**BE**		
		I	**II**	**III**
3.3	erläutern Eine Störung wird häufig über ein LOW-Signal gemeldet, da eine Signalunterbrechung auf diese Weise ebenfalls erfasst wird. Ohne Störung wird ein HIGH-Signal gesendet. Wird nun die Leitung unterbrochen, so kann das HIGH-Signal nicht mehr ankommen.	3		

3.4	implementieren
	
	SOW_CURVE:
	sbis 0x09,2 //Port D ist Eingangsport. Bit 2 erhält das
	//Anforderungssignal
	//für die Kurvenfahrt. Die Schleife wird unterbrochen, wenn
	//Kurvenfahrt nicht mehr angefordert wird.
	RET //Ende des Unterprogramms und Fortsetzung des Hauptpr.
	sbi 0x05,0 //Ausgänge für Motor laut Struktogramm setzen
	sbi 0x05,1 //Anlehnung an den Schrittmotor
	cbi 0x05,2
	cbi 0x05,3
	call WAIT_10ms // Wartezeit vor dem erneuten Setzen der Ausgänge
	cbi 0x05,0
	sbi 0x05,2
	call WAIT_10ms
	cbi 0x05,1
	sbi 0x05,3
	call WAIT_10ms
	sbi 0x05,0
	cbi 0x05,2
	call WAIT_10ms
	jmp SOW_CURVE //Wiederholung der Motordrehung, bis nicht geskippt wird
			3	3

3.5	erläutern Das Niveau könnte mit einem Lichtsensor ermittelt werden. Ist dieser bedeckt, so befindet sich noch Saatgut im Behälter. Ohne Saatgut fällt z. B. Licht auf ein Photoelement des Sensors. Dadurch wird erkannt, dass das Niveau unterschritten ist.	3		
	Summe 26	**6**	**11**	**9**

III Bewertung und Beurteilung

Die Bewertung und Beurteilung erfolgt unter Beachtung der nachfolgenden Vorgaben nach § 33 der Oberstufen- und Abiturverordnung (OAVO) in der jeweils geltenden Fassung. Bei der Bewertung und Beurteilung der sprachlichen Richtigkeit in der deutschen Sprache sind die Bestimmungen des § 9 Abs. 12 Satz 3 OAVO in Verbindung mit Anlage 9b anzuwenden.

Bei der Bewertung und Beurteilung der Übersetzungsleistung in den Fächern Latein und Altgriechisch sind die Bestimmungen des § 9 Abs. 14 OAVO in Verbindung mit Anlage 9c anzuwenden.

Der Fehlerindex ist nach Anlage 9b zu § 9 Abs. 12 OAVO zu berechnen. Für die Ermittlung der Punkte nach Anlage 9a zu § 9 Abs. 12 OAVO sowie Anlage 9c zu § 9 Abs. 14 OAVO wird jeweils der ganzzahlige nicht gerundete Prozentsatz bzw. Fehlerindex zugrunde gelegt.

Für die Bewertung in den modernen Fremdsprachen ist der „Erlass zur Bewertung und Beurteilung von schriftlichen Arbeiten in allen Grund- und Leistungskursen der neu beginnenden und fortgeführten modernen Fremdsprachen in der gymnasialen Oberstufe, dem beruflichen Gymnasium, dem Abendgymnasium und dem Hessenkolleg“ vom 7. August 2020 (ABl. S. 519) zugrunde zu legen. Demnach erfolgt die Bewertung und Beurteilung mit der Maßgabe, dass lediglich bei der Ermittlung des Prüfungsergebnisses (Note) aus Prüfungsteil 1 und 2 gerundet wird.

Darüber hinaus sind die Vorgaben der Erlasse „Hinweise zur Vorbereitung auf die schriftlichen Abiturprüfungen (Abiturerlass)“ und „Durchführungsbestimmungen zum Landesabitur“ in der für den Abiturjahrgang geltenden Fassung zu beachten.

Als Kriterien für die Bewertung und Beurteilung dienen unter Beachtung der Zielsetzung der gymnasialen Oberstufe nach § 1 Abs. 2 OAVO neben dem Inhaltlichen auch die in den Kerncurricula genannten überfachlichen Kompetenzen, insbesondere die Sprachkompetenz und Wissenschaftspropädeutik; dies zeigt sich u.a. in qualitativen Merkmalen wie Strukturierung, Differenziertheit, (fach-)sprachlicher Gestaltung und Schlüssigkeit der Argumentation.

Im Fach Technische Informatik besteht die Prüfungsleistung aus der Bearbeitung eines Vorschlags, wofür insgesamt maximal 100 BE vergeben werden können. Ein Prüfungsergebnis von **5 Punkten (ausreichend)** setzt voraus, dass mindestens 45% der zu vergebenden BE erreicht werden. Ein Prüfungsergebnis von **11 Punkten (gut)** setzt voraus, dass mindestens 75% der zu vergebenden BE erreicht werden.

Gewichtung der Aufgaben und Zuordnung der Bewertungseinheiten zu den Anforderungsbereichen

Aufgabe	**Bewertungseinheiten in den Anforderungsbereichen**			**Summe**
	AFB I	**AFB II**	**AFB III**	
1	16	18	16	**50**
2	8	14	2	**24**
3	6	11	9	**26**
Summe	**30**	**43**	**27**	**100**

Die auf die Anforderungsbereiche verteilten Bewertungseinheiten innerhalb der Aufgaben sind als Richtwerte zu verstehen.